VNF PACKAGE SPECIFICATION GS NFV-SOL004

Andrei Kojukhov, PhD - VNF Package Spec Rapporteur, Amdocs

Background

- Security sub-committee have been studying the security of the VNF package.
- 3 priorities have been identified:
  - **Priority 1: VNF Package Verification**
    - Integrity of the VNF package needs to be verified prior to, or at the time of onboarding. The purpose is to ensure that the VNF package originates from the vendor, and that the content has not been tampered with. The verification is done against the signature provided by the vendor. Reference [ETSI NFV SOL004] contains the detailed specifications.
  - **Priority 2: Integrity Verification at Instantiation**
    - At instantiation, the integrity of VNF image and related files shall be verified. The options are:
    - A) Verify against the signature provided by the vendor. [ETSI NFV SOL004] specifies "*The VNF provider may optionally digitally sign some artifacts individually*".

      B) Verify against the signature created by the service provider. [ETSI NFV SOL004] specifies "*If software images or other artifacts are not signed by the VNF provider, the service provider has the option, after having validated the VNF Package, to sign them before distributing the different package components to different function blocks or the NFVI*".
  - **Priority 3: Service Provider Ability to Sign the Artifacts**
    - If the vendor did not sign artifacts (inside the VNF package) individually, service provider may want to sign those. Also, if the service provider needs to modify or add any artifacts, the service provider may want to sign those.

ONAP Security Sub-committee Recommendation

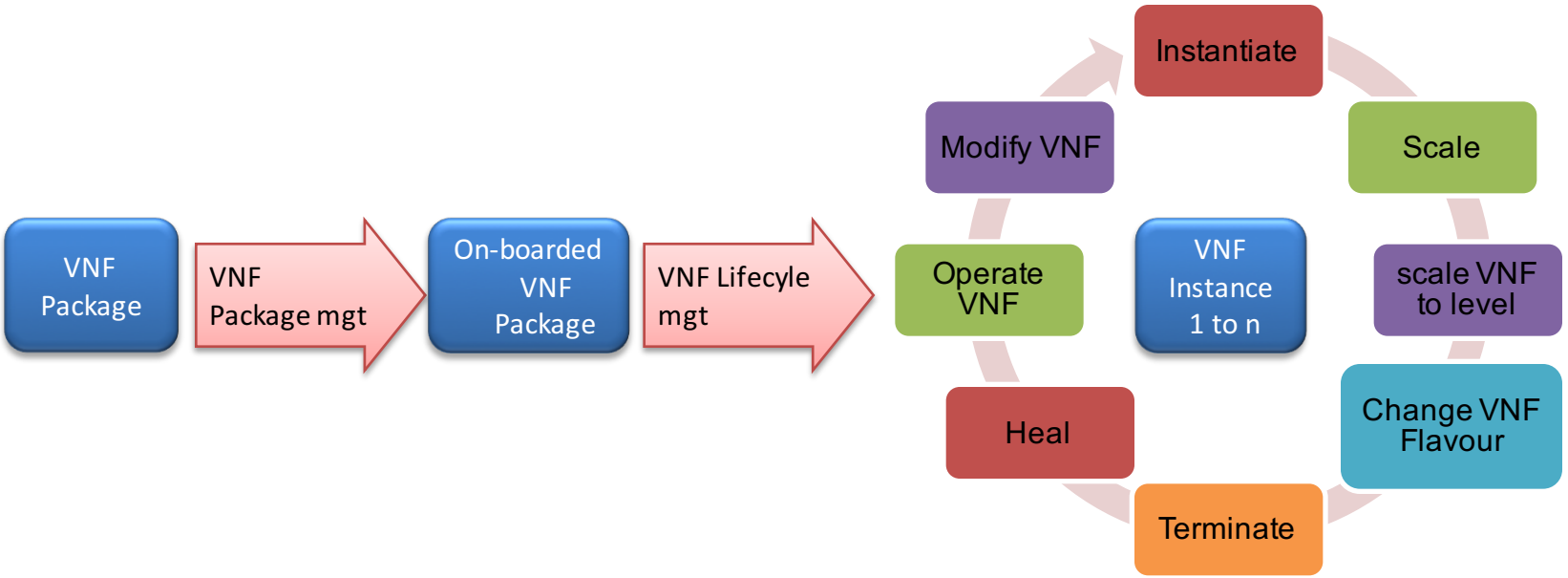
- For Beijing, only priority 1 is brought forward for recommendation at this time.
  - VNF package integrity check at onboarding (VNF SDK and/or SDC) to follow VNF package specification as defined in SOL004.

- Note: This assumption aligns with the VNF SDK assumptions
  - VNF-SDK User Story: VNFSDK-170

- Request to the TSC: Endorse the recommendation that VNF package integrity check at onboarding follows the VNF package specification SOL-004.
  - Other use cases are FFS and to be brought forward later.

VNF Package Onboarding

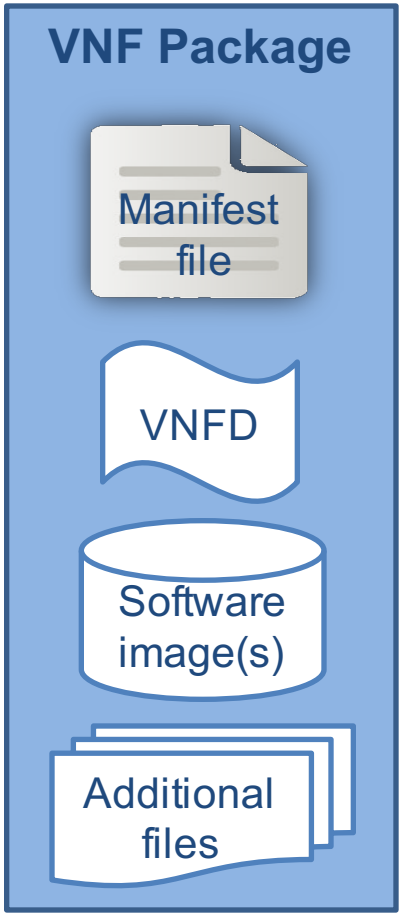
- On-boarding a VNF package is a pre-requisite to VNF instantiation and lifecycle management



Note: see next slide for detail of all operations

VNF Package Structure Defined in SOL004

- The VNF Package contains:
  - the VNF descriptor (VNFD) that defines metadata for package onboarding and VNF management,
  - the software images needed to run the VNF, and
  - Manifest file that provides package integrity and authenticity
  - (optional) additional files to manage the VNF (e.g. scripts, vendor-specific files etc.).
- The VNF Package is delivered by the VNF provider as a whole and is immutable (protected from modification).
- The VNF Package or its Manifest file is digitally signed
- The VNF Package is stored in a repository by the NFVO.
- The VNF Package can be then accessed by VNFM.



Reference:
- ETSI GS NFV-IFA 011
- ETSI GS NFV-SOL 004

VNF Package format

- A VNF Package is a Cloud Service ARchive (CSAR)
- A CSAR file is a ZIP file with a well-defined structure.
  - The structure and format of a VNF package conform to the TOSCA Simple Profile YAML v1.1/v1.2 Specification of the CSAR format.
- The VNFD is the main TOSCA definitions YAML file inside the archive.

References:
- TOSCA-Simple-Profile-YAML-v1.1

VNF Package Option 1 – used in ONAP Amsterdam - CSAR with Metadata File

- The TOSCA.meta file includes block_0 with the Entry-Definitions keyword pointing to a TOSCA definitions YAML file used as entry for parsing the contents of the overall CSAR archive – MRF.yaml

  TOSCA-Meta-File-Version: 1.0

  CSAR-Version: 1.1

  Created-by: Company Name

  Entry-Definitions: Definitions/ MRF.yaml

- Any TOSCA definitions files besides the one denoted by the Entry-Definitions can be found by processing respective imports statements in the entry definitions file (or in recursively imported files)

- Any artifact files (e.g. scripts, binaries, configuration files) can be either declared explicitly through blocks in the TOSCA.meta file or pointed to by relative path names through artifact definitions in one of the TOSCA definitions files contained in the CSAR file.

```
!------TOSCA-Metadata
      !------TOSCA.meta

!------Definitions
      !----- MRF.yaml
      !----- OtherTemplates (e.g.,
             type definitions)

!------Artifacts
      !----- ChangeLog.txt
      !----- MRF.cert
      !----- image(s)
      !------Tests
      !------Licenses
      !------Scripts
        !----- install.sh

!----- MRF.mf
```

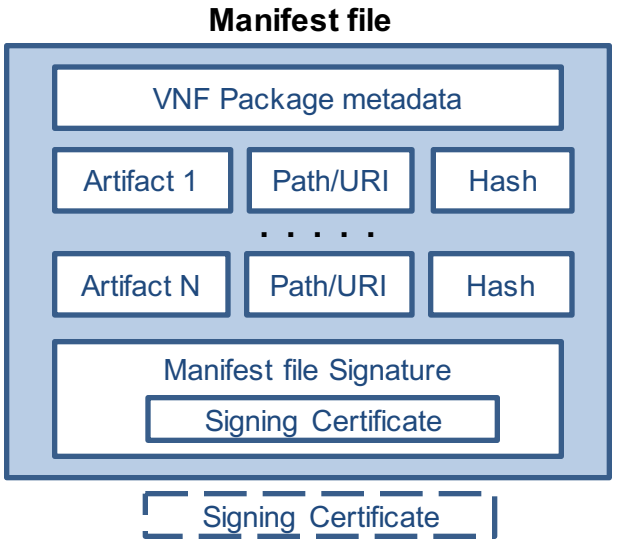
References:
- ETSI GS NFV-SOL 004
- TOSCA-Simple-Profile-YAML-v1.1

VNF Package Standard Artifacts and Directories

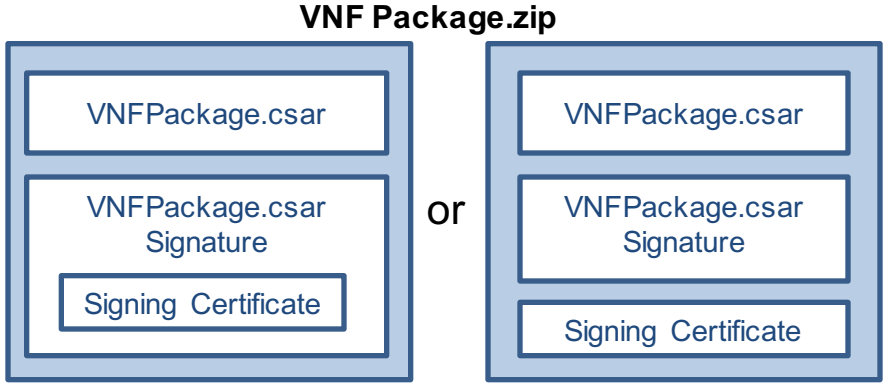
- Change History File
  - Humanly readable text file
  - All the changes in the VNF package shall be versioned, tracked and inventoried
- Testing Files
  - Goal is to enable VNF package validation
  - VNF Provider includes files containing necessary information (e.g. test description)
- Licensing Information for released VNF
  - Include a single license term for the whole VNF.
  - In addition may include license terms for each of the VNF package artifacts if different from the one of the released VNF
- A directory for artifacts serving 3-rd party extensions
  - Allow adding artifacts supporting external (non-MANO) extensions e.g. ONAP extensions

VNF Package Integrity – ONAP Priority 1

- Option 1: **Manifest file** - based if there are both local and external artifacts
  - A Digest hash per each artifact
  - Manifest file is signed with VNF provider private key
  - VNF provider's certificate includes a VNF provider public key
  - The certificate may be a separate artifact or included in the signature container, e.g. CMS



- Option 2: **CSAR**-based if all artifacts are located inside a CSAR
  - CSAR file is digitally signed with the VNF provider private key
  - VNF provider delivers one zip file containing a CSAR file, a signature file and a certificate file that includes a VNF provider public key
  - The certificate may be a separate artifact or included in the signature container, e.g. CMS



- Both options rely on existence in the NFVO of a root certificate of a trusted certificate authority, delivered via a trusted channel separately from a VNF package

Manifest File – ONAP Priority 1

- VNF package **metadata**

- A list of blocks each is related to one file in the VNF package, including
  - **Source**: artifact URI
  - **Algorithm**: name of an algorithm used to generate the hash
  - **Hash**: text string corresponding to the hexadecimal representation of the hash

- **Manifest file Signature**

metadata:
vnf_product_name: vMRF-1-0-0
vnf_provider_id: Acme
vnf_package_version: 1.0
vnf_release_data_time: 2017.01.01T10:00+03:00

Source: MRF.yaml
Algorithm: SHA-256
Hash: 09e5a788acb180162c51679ae4c998039fa6644505db2415e35107d1ee213943

Source: scripts/install.sh
Algorithm: SHA-256
Hash: d0e7828293355a07c2dccaaa765c80b507e60e6167067c950dc2e6b0da0dbd8b

Source: https://www.vendor_org.com/MRF/v4.1/scripts/scale/scale.sh
Algorithm: SHA-256
Hash: 36f945953929812aca2701b114b068c71bd8c95ceb3609711428c26325649165

-----BEGIN CMS-----
MIGDBgsqhkiG9w0BCRABCaB0MHICAQAwDQYLKoZIhvcNAQkQAwgwXgYJKoZIhvcN
AQcBoFEET3icc87PK0nNK9ENqSxItVIoSa0o0S/ISczMs1ZIzkgsKk4tsQ0N1nUM
dvb05OXi5XLPLEtViMwvLVLwSE0sKIFIVHAqSk3MBkkBAJv0Fx0=
-----END CMS-----

References:
- IANA register for Hash Function Textual Names
  https://www.iana.org/assignments/hash-function-text-names/hash-function-text-names.xhtml

Signing Individual Artifacts – ONAP Priority 2

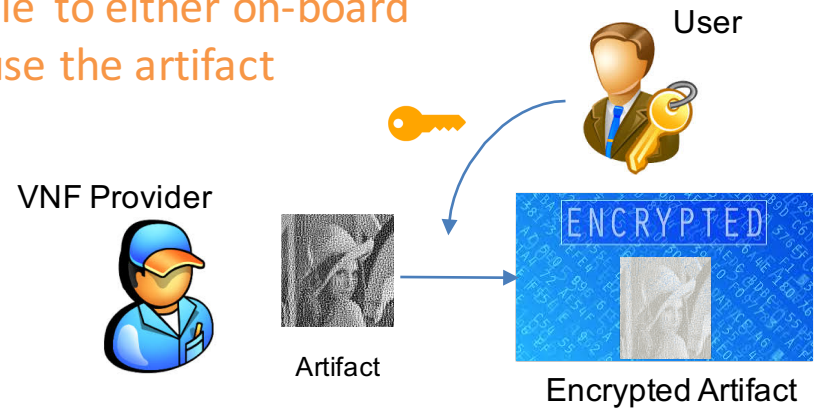
- VNF provider may sign individual artifacts adding a signature file in standard format (e.g. CMS, PKCS#7)
- A certificate file with extension .cert accompany the signed artifact
- The signature and certificate files have the same name and location as the signed artifact
  - If the signature format allows it, the certificate may be included in the signature file

```
!--------- Artifacts
        !----- install.sh
        !----- images
                !----- MRF.img
                !----- MRF.cert
                !----- MRF.cms
```

Encrypting Security Sensitive Artifacts - Not considered yet by ONAP

A public key is provided by the party who is responsible to either on-board the package or use the artifact

- Asymmetric encryption:
  - VNF provider uses the user public key to encrypt the security sensitive artifact
  - A consumer of the artifact then decrypts the artifact with its own private key



- Symmetric encryption:
  - The artifact is encrypted with the VNF provider key shared with the consumer in encrypted form (a user key is used for encryption)
  - A consumer of the artifact decrypts the shared key with its own private key and then uses the obtained shared key to decrypt the artifact
  - The encrypted artifact is delivered in a CMS file, with all info needed to decrypt it: algorithm used for artifact encryption, encrypted key used for artifact encryption and algorithm used to encrypt the key.